All-En-Route Cab Aggregator

2023

Table of Contents

**Project Description2**

Purpose and scope2

Stake-holders and resposibilities2

Service availability and accessibility requirements2

Service platform entities and functional requirements2

Service platform quality attribute requirements2

**Functional Architecture and Architecture Principle3**

Functionality for passengers3

Functionality for cab providers5

Functionality for cab aggregator (buiseness owners)7

Functionality for system administrator7

Functionality for customer support personnel8

Functionality for technical support personnel8

Functionality for kiosks operator9

**Software Architecture10**

Software overview10

User applications10

Rider application11

Driver application14

Operations application16

Core and platform services18

Integration Architecture23

**Deployment Architecture28**

Deployment overview28

Deployment topology28

Deployment components29

Deployment process32

Security considerations33

Monitoring and logging33

Disaster recovery and backup33

**References34**

**All-En-Route – Cab Aggregator Service**

**1. Purpose and Scope**

The Indian cab market, valued in thousands of crores (INR), is experiencing expansion, and organized cab aggregators are progressively strengthening their position in the market.

**All-En-Route** intends to provide a **nationwide online cab aggregator service** that is **safe**, **secure**, **accessible at any time**. It aims to **generate revenue** by facilitating **mutually-beneficial optimal matches** between registered passenger **commute demands** and registered **cab service providers**.

It will offer online **ride-booking services** for both intra and inter-city travel, employing **multiple drivers** for l**ong-distance rides**. Additionally, it will integrate with food delivery service partners to provide **on-the-go food ordering services** for passengers during their journeys.

**2. Stake-holders and Responsibilites**

Across the nation, cab providers and passengers willing to avail the services of this platform will need to get registered using the All-En-Route application.

**Passengers**, once registered successfully, will use the application to manage their profiles, book, update, track, or cancel rides (with valid reasons), order food for their journey, and make payments for completed rides.

**Cab-providers**, once approved and registered successfully, will use the application to manage their profiles, update cab and driver information, select from available ride requests, update ride start and completion statuses, and cancel selected rides (with valid reasons).

**Administrators** will use the application to verify, approve and notify cab providers and food delivery partners as well as manage the platform data and resources.

**Support personnel** will use the application to address customer requests and concerns.

**3. Service Availability and Accessibility Requirements**

All-En-Route will be **accessible** from **anywhere in the country** and **available 24x7** for all above-mentioned key stake-holders of the service.

Service kiosk will be made available at heavy-footfall locations like airports, railway stations, shopping malls and markets.

**4. Service Platform Entities and Functional Requirements**

The All-En-Route service platform primarily manages **passenger profiles**, **passenger location data**, **cab and driver profiles**, **cab location data**, **ride details** and relevant **food order details** (if applicable) along with **payment status**.

The service platform will **optimally match** the passenger **ride requests** (from current or pickup location) with **nearby** available **cabs**, following **applicable business rules** and **constraints**. The **long-distance rides** will follow a maximum at-a-stretch driving time restriction for drivers and apply the **multi-driver strategy** to enable attentive driving and ensure safe journeys.

The **service platform** will **annotate ride routes with predetermined food delivery points** along the way, considering distance and/or time factors. **Passengers** **can** then **choose to order food from the menu of one of these designated delivery locations** at their discretion.

Functionally, the service platform will maintain passenger profiles, cab provider profiles, offer mapping services, provide information about cab availability, pricing, and routes. It will also enable passengers to select their source and destination and optionally order food for their ride and enable cab providers to select from available ride requests. Additionally, the platform will track cab locations before and during the ride, track details and statuses of ongoing and scheduled rides, oversee payment statuses. Moreover, the platform will archive records of rides, food orders, and payment transactions for a specified period. The service will also incorporate regular updates and checks to ensure compliance with regulatory requirements such as vehicle standards, driver qualifications, fares regulations, safety measures, and data sharing.

Furthermore, the service platform will **integrate** with **verified food delivery partner services** to provide on-the-go food delivery service for passengers. Additionally, it will **integrate** with **various payment services** for managing ride payments and with **reliable third-party platform** for maps, routes and location services.

**5. Service Platform Quality Attribute Requirements**

For safety and user convenience, the service platform will provide cab location tracking before and during the ride, and users can choose to share their location. It will be accessible nationwide around the clock and the client-facing operations will be highly responsive with minimal latency. Additionally, to safeguard user privacy, any personally identifiable information (PII) must not be publicly accessible and all transaction data will be encrypted.

**All-En-Route Functional Architecture and Architecture Principles**

All-En-Route service platform will provide functionality for passengers, cab providers, business owner, management personnel, support personnel and kiosk operators. This will include user management, mapping services (map location and route), ride management, enroute food delivery, payment processing, business analytics, business management, platform management, administration, and customer (passenger and cab providers) support.

It will also take into consideration privacy and security, availability, responsiveness, performance, usability, and maintainability requirements that are essential for its promised functionality. The specifics of both these aspects are described in the document below.

**1. Functionality for Passengers**

**1.1 Account management**

Account management will provide registration, authentication, and profile management functionality.

For better **usability** or user experience, usual standard sign-up, login, logout, and profile management processes will be used.

**1.1.1 Registration**

After installing All-En-Route client, for registration process, passenger will need to enter name, user ID, phone, e-mail and optionally address details. The phone will be verified using OTP and e-mail will be verified using verification link. On successful verification, the Passenger details will be persisted in the registered passenger’s database, where Personally Identifiable Information (PII) details will be encrypted before persisting. The passenger will then be presented with user interface to setup password. On successful setup, password will also be encrypted and persisted.

PII details and password will consistently be encrypted while in-transit as well as at-rest for **privacy** and **security** reasons.

**1.1.2 Authentication**

Successfully registered passengers will be able to login into the service platform using user ID and password credentials. The login credentials will be validated against registered passenger’s database. On successful login, passengers will be presented with ride booking interface. On failure, error message will be displayed followed by login interface to reattempt login. Once logged-in, logout will take the logged-in passenger back to the login interface.

Credentials will be encrypted while in-transit for **privacy** and **security** reasons.

**1.1.3 Profile management**

Passenger will be able to setup and update the profile – for home, office and favorite locations addresses and payment methods, if any.

**1.2 Mapping Services**

The mapping services will provide for map view, location, route information and location tracking.

The mapping services must be **responsive** and should have low latency (< 10 seconds).

All-En-Route will integrate with a mapping service partner to provide location and route information. The interaction with the mapping service will be done by calling its platform interfaces.

**1.2.1 Locations and routes**

Ride booking interface will provide map view for locations and routes. Map view will display source (current or entered) and destination locations as markers on the map with route from source to destination. The platform will use reliable third-party mapping service interface. History of last 10 searches for destination will be made available for better **usability**.

**1.2.2 Location tracking**

Cab location before ride: The passenger will be able to track the assigned cab’s location starting 5 minutes prior to the ride's scheduled start time.

Cab location during ride: The passenger will be able to track the cab’s location during the ongoing ride. For **safety** reasons, passenger will also be able to share it with family or friends.

**1.3 Ride management**

Ride management functionality for passengers will provide for ride request creation, ride assignment notification and ride cancellation.

**Availability** of ride management services and mapping services (mentioned in 1.2 above), will be essential for passengers to be able to book rides **anytime anywhere across the country**.

The ride management operations must be **responsive** and should have low latency (< 30 seconds). Long running operations like ride assignment should keep the passenger updated of the ongoing status.

**1.3.1 Ride request creation**

Passengers will be able to use the mapping service to setup ride request by selecting source and destination locations, and optionally selecting number of passengers and luggage capacity required. The service platform will present route along with available cab types and pricing for the ride to the passenger. The passenger will then select from available cab types and pricing, set the ride time to now or schedule for later and create the ride request.

**1.3.2 Ride assignment and notification**

**1.3.2.1 Regular ride assignment**

The service platform will match available ride requests with available cabs with a timeout of 3 minutes applying location proximity starting from 0.5 km radius with an increase of 0.5 km every minute. The available cabs matching process will also incorporate passenger preferences selected, if any, mentioned in 1.3.1 above.

**1.3.2.2 Long-distance-multi-driver ride assignment**

The long-distance rides will follow regulatory requirements, if any. Rides estimated to be more than 4 hours will be treated as long distance rides. For long distance rides, after every 4 hours change of driver will be mandatory. The ride will be broken into segments of 4 hours with estimated start and end time-ranges and all ride segments will be assigned to appropriately matched cab providers. The best-effort ride request matching (minimizing cab switches) will have timeout of 10 minutes. Passenger will have an option to select “best-effort processing with 10 minutes wait time”, and if not selected, normal matching with 3 minutes timeout will be performed.

**1.3.2.3 Ride assignment notification**

Once the ride is assigned, the service platform will notify the ride OTP and details of cab and driver to the passenger who requested that ride. The ride will be maintained in the upcoming rides for the passenger before start of the ride. The passenger will receive updates about changes, if any, regarding the assigned cab and driver. The passenger will be able to track the cab's location starting 5 minutes prior to the ride's start time.

**1.3.3 Ride cancellation**

Passenger will be allowed to cancel the ride request before it is assigned. In addition, passenger will be allowed to cancel the assigned ride 30 minutes before the scheduled start time for regular rides and 2 hours before the scheduled start time for long distance rides. Along with these time constraints, the cancellation must be accompanied with a valid reason from a pre-defined list. Otherwise, any ride cancellations will result in penalty of INR 200 or 10% of ride pricing, whichever is higher.

On cancellation of long-distance rides by passenger, the driver marks the finish of the current segment of the ride and passenger will be charged for completed segments of the ride along with delivered food orders (if any) and the penalty mentioned above.

**1.4 Enroute food delivery option**

The enroute food delivery service will only be available for long distance rides. The service platform will provide passengers with available food delivery stops (estimated time), food menu and delivery partner options available for their upcoming or ongoing long-distance ride route. The passengers will be able to choose food stop(s) and order food for enroute delivery.

All-En-Route will integrate with three food delivery service partners to fulfill the enroute food delivery service for the passengers. The interaction with the food delivery services will be done by calling their platform interfaces. The food delivery personnel will not access the All-En-Route service platform.

**1.4.1 Food stops and options listing for ride route**

Once the ride begins, the service platform will use mapping service interface to mark locations spanned at one-hour intervals from ride start time. The service platform will then use the food delivery partner platform interface to query availability of its food delivery service at each of these hourly locations. If there is a food delivery service available at an hourly location, it will be marked as a food stop. It will also determine the estimated arrival time-range of the cab at each food stop by adding 20 minutes (+/- 10 minutes) buffer. It will then query the interface of food delivery partner(s) for food menu(s) at each of the food stop locations.

It will provide passenger with this information of available food options and food stops, for getting enroute food delivery for their upcoming or ongoing rides.

**1.4.2 Food order delivery functionality for ongoing ride**

The passenger will select food from available options and order the food for delivery at specific food stops. The service platform will send the food delivery order to the food delivery partner using their interface and notify the passenger with confirmation status received. The delivered food order bill amount will be added to the ride cost.

If the food delivery service does not meet the specified delivery time-range, it will be imposed with a penalty of 5% of the food bill by All-En-Route. On the other hand, if the cab fails to arrive in a timely manner to pick up the food, All-En-Route will bear the 5% penalty of the food bill to compensate the food delivery partner.

**1.5 Payments management**

Passenger client interface will have the usual standard payment processing options available, as mentioned in the subsections below. Only digital payment options will be provided, cash payment will not be accepted.

Payment method details entered by passenger will not be stored and will be masked when displayed in the interface with masking characters (‘\*’ or ‘x’) for **privacy** reasons. All the payment transaction records will be encrypted while in-transit as well as at-rest for **security** reasons.

**1.5.1 Ride payments**

The payment will need to be done once the ride is marked finished. Any pending penalty amounts (not yet paid by passenger) will be recovered along with the next ride payment.

UPI, digital wallet, net banking, credit/debit cards options will be made available to the passenger for the ride payment.

After successful payment, passengers will receive the invoice for their ride and penalty payments on their registered email.

**1.5.2 Food delivery order payments**

The payments for food delivery order(s) will be added to ride cost. Payments to food delivery partners will be settled by the service platform on daily basis.

**2. Functionality for Cab providers**

**2.1 Account management**

Account management will provide onboarding, authentication, and profile management functionality. For cab providers, onboarding will have regulatory verification requirements. The cab providers will need to sign contract agreeing to the terms and conditions specified in the contract.

For better **usability** or user experience, usual standard login, logout, and profile management processes will be used.

**2.1.1 Onboarding**

After installing All-En-Route client, for registration process, cab provider will need to enter driver details – name, user ID, phone, e-mail, address details, birth date, Aadhaar card, driving license, and health record details. Cab providers will also need to provide cab details – cab registration certificate, tax payment record, financial NoC (if applicable), cab type, seating and luggage capacity, cab maintenance records. The phone will be verified using OTP and e-mail will be verified using verification link. The rest of the driver and cab information will be verified by the All-En-Route system administrator as per regulatory requirements and approved accordingly. The cab providers will then need to sign contract agreeing to the terms and conditions specified. On successful verification and contract signing, cab and driver details will be persisted in registered cab providers database, where Personally Identifiable Information (PII) details will be encrypted before persisting. The cab provider will then be presented with interface to setup password. On successful setup encrypted password will also be persisted.

PII details and password will consistently be encrypted while in-transit as well as at-rest for **privacy** and **security** reasons.

**2.1.2 Login and logout**

Successfully onboarded cab providers will be able to login into the service platform using user ID and password credentials. The login credentials will be validated against onboarded cab providers database. On successful login, cab providers will be presented with interface to select and accept from available ride requests. On failure, error message will be displayed followed by login interface to reattempt login. Once logged-in, logout will take the logged-in cab provider back to the login interface.

Credentials will be encrypted while in-transit for **privacy** and **security** reasons.

**2.1.3 Profile management**

Cab providers will need to enter and be able to update their bank details to receive ride payments. Cab providers will also be able to update driver details – driving license and health records, as well as cab maintenance records and change in cab details, if any.

**2.2 Mapping Services**

Cab providers will be provided with map view of assigned ride source (passenger pickup location) and destination along with route and real time driving directions.

The mapping services must be **responsive** and should have low latency (< 10 seconds).

**2.3 Ride management**

Ride management functionality for cab providers will provide for ride assignment, ride processing and cancellation.

**Availability** of ride management services and mapping services (mentioned in 2.2 above), will be essential for cab providers to get rides assigned and be able to process the rides assigned **anytime anywhere across the country**.

For efficient ride assignments and processing, ride management and mapping services operations will need to be **responsive** (low latency).

**2.3.1 Ride request assignment**

**2.3.1.1 Regular ride request matching**

The service platform will match available ride requests with available cabs with a timeout of 3 minutes applying location proximity starting from 0.5 km radius with an increase of 0.5 km every minute. The available cabs matching process will also incorporate passenger preferences selected, if any, mentioned in 1.3.1 above.

**2.3.1.2 Long-distance-multi-driver ride request matching**

The long-distance rides will follow regulatory requirements, if any. Rides estimated to be more than 4 hours will be treated as long distance rides. For long distance rides, after every 4 hours change of driver will be mandatory. The ride will be broken into segments of 4 hours with estimated start and end time-ranges. These segment ride requests will have start time-range and end time-range instead of fixed time. The start time-range and end time-range will use 20 minutes (+/- 10 minutes) time buffer to accommodate early or delayed arrival from previous segment of the ride. During matching, to minimize cab switches, preference will be given to the cab providers who can pick all or most of the consecutive segments of the ride with required mandate of change of driver per segment of the ride. For the ride segments not picked as consecutive segments, matches will be performed just like the “regular ride matching” mentioned above in 2.3.1.1. In all conflict cases, first-come-first-serve criterion will be applied. The best-effort matching (minimizing cab switches) if opted for by passenger, will have timeout of 10 minutes. If passenger did not opt for best-effort matching, normal matching with 3 minutes timeout will be performed.

**2.3.1.3 Ride request assignments**

The matched cab providers will then be notified with the available ride requests (includes ride segments in case of long-distance rides). The cab provider will be able to select and accept one from available ride requests. The service platform will then assign the ride request to the cab provider using criteria as mentioned in 2.3.1.2. The cab provider will be then notified of ride assignment. The cab will be marked busy from the beginning of the ride until 5 minutes before the end time.

**2.3.2 Ride processing**

**2.3.2.1 Regular ride request processing**

For regular rides, the cab driver will mark the start of the ride using start OTP provided by the passenger and mark the finish of the ride, once the ride is completed, to trigger the payment process.

**2.3.2.2 Long-distance-multi-driver ride request processing**

In case of long-distance rides, the change of driver will happen after the segment of ride is completed by the current driver. A change of cab may also be needed if different cab providers are serving consecutive ride segments. Each driver will mark the finish for the segment of the ride completed by that driver. For the last segment of a long-distance ride, the segment finish will also mark the finish for the ride to trigger the payment process.

**2.3.3 Ride cancellation**

The cab provider will be allowed to cancel the ride 30 minutes before the scheduled start time for regular ride and 2 hours before the scheduled start time for long distance ride. Along with these time constraints, the cancellation must be accompanied with a valid reason from a pre-defined list. Otherwise, any ride cancellations will result in penalty of INR 200 or 10% of ride (segment) pricing, whichever is higher.

For long-distance rides, if cab provider cancels assigned segment(s) of the ride, the cab provider will be charged with the penalty mentioned above and All-En-Route service platform will replace the cab provider by performing prioritized re-matching of the cancelled segment(s) of the ride. The passenger will be kept informed of the replacement process status. If no replacement is found within 20 minutes of start of the cancelled segment, the passenger will have right to cancel rest of the ride segments without any penalty and All-En-Route will compensate the passenger with discount of 20% of remaining ride cost, for future rides.

**2.4 Payments management**

Payments to cab providers will be settled by the service platform on daily basis. The cab providers will receive consolidated statement for the rides completed and penalties (if any) for the day, on their registered email.

All the payment transaction records will be encrypted while in-transit as well as at-rest for **security** reasons.

**3. Functionality for cab aggregator (business owner)**

The business owner account will be pre-configured in the system and the business owner will need to change the password on first login. The service platform will provide executive dashboard for business owner. The business owner will be able to configure the dashboard with the required basic and analytical charts for aggregated metrics, dimensions, and time granularities of interest. The charts will be created using the outputs of the queries run against the archived data of the service platform.

The aggregated metrics (business measures) will include number of registered passengers, number of onboarded cab providers, rides served, and revenue numbers (total revenue, profit/loss). The dimensions will include locations (city or region), time-range of the day, cab type (hatchback, sedan, SUV), ride type (regular, long distance). The time granularities will include day, week, month, quarter, year and custom time range.

The executive dashboard functionality must provide for **usability** (user-friendly interface) and **configurability**.

**3.1 Basic charts**

The basic charts functionality will allow the business owner to choose the metrics, time granularity for metric aggregation and dimensions to create charts of interest. It will be a two-dimensional chart of aggregated metrics vs dimensions.

**3.2 Analytics charts**

The analytics charts functionality will allow the business owner to create comparison (previous vs current quarter/year) charts and trend (most recent 12 months/4 quarters/5 years) charts.

It will also allow exploring the basic charts on an additional second dimension. This will enable the business owner to conduct a two-dimensional analysis of the business. For example, this will allow the business owner to explore the basic total-revenue-by-location chart further for specific locations on a second dimension such as cab-type.

**3.3 Contracts and compliance information**

The dashboard will also provide information on regulatory compliance statuses, business licenses with validity time frames, long-term partner service subscriptions or contract fees and renewal due dates.

Based on the insights collected from the analytical summaries presented in the charts, business owner will be able to make high level business decisions and set targets for quarterly and yearly business growth and revenues, which will then be communicated to the management and business development teams outside of the cab aggregation service platform.

**4. Functionality for system administrator**

The system administrator account will be pre-configured in the system and the system administrator will need to change the password on first login and will update the profile information.

The functionality for system administrator will include user (support, kiosk operator, cab provider) account management, service platform management (installs, configurations, upgrades and uninstalls).

**4.1 User account management**

The functionality will provide create, view, update and delete functionality for support users and kiosk operator accounts for the service platform.

Using this functionality, cab provider accounts reported to be non-compliant (by automated periodic regulatory checks) will be blocked and marked unavailable by system administrator. These cab providers will be notified and provided with grace period to comply with regulatory requirements before their accounts are permanently removed from the system by system administrator.

**4.2 Service Platform Installation Management**

This functionality will allow system administrator to install software on initial (or additional) hardware, apply required software upgrades, bug fixes and security patches for the service platform as well as uninstall if required.

The configuration functionality will include configuration of alert conditions for service platform health and resource capacity monitoring, configuration of frequency for automated periodic regulatory checks and configuration of data archival based on time limit.

The service platform will need to support **deployability** and **maintainability** with non-disruptive scaling and upgrades.

**5. Functionality for customer support personnel**

Customer (passenger/cab provider) issues will be handled by the customer support. The customer support personnel will be the first level of support and unresolved issues will be passed on to the second level of support handled by the technical support personnel.

The customer support personnel accounts will be created by system administrator. The usual standard login, logout and password update functionality will be provided to the customer support personnel.

**5.1 Customer issue processing**

Customer support personnel will have access to ongoing and completed rides and food orders from the current day as well as a one-week history of completed rides and food orders, to be able to address issues reported by passengers and cab providers.

Customer reported issues from service platform will be added to the current list of customer issues sorted with severity (high, medium, low) level assigned by customer. The issues will get assigned to customer support personnel based on their current workload. Customer support personnel will be able to view assigned issues, update it with resolution and mark it as resolved or forward it to technical support. Every change of status and resolution of the issues will be notified to respective customers.

**5.2 Support knowledge base search**

Customer support personnel will have interface to search knowledge base (KB) for the reported customer issues. On success, the issue will be resolved with the known resolution provided in the KB. Otherwise, the issue will be forwarded to technical support for further investigation and resolution.

Additionally, there is the potential to utilize chatbot and virtual assistant functionalities in customer support, which will then need to be responsive.

**6. Functionality for technical support personnel**

Issues that could not be resolved by customer support personnel will be forwarded to the technical support.

The technical support personnel accounts will be created by system administrator. The usual standard login, logout and password update functionality will be provided to the technical support personnel.

**6.1 Customer issue processing**

Technical support personnel will have access to ongoing and completed rides, food orders, customer feedback, and driver ratings, to be able to analyze and address customer (passenger/cab provider) issues.

Forwarded issues will be added to the current list of customer issues sorted with severity (high, medium, low) level assigned by customer. The issues will get assigned to technical support personnel based on their current workload. Technical support personnel will be able to view assigned issues, analyze and resolve the issues. Every change of status and resolution of the issues will be notified to respective customers. The issues that need a change in the service platform will be resolved as “will be addressed in upcoming upgrade or patch process” with more information if available at that point in time.

**6.2 Support knowledge base update**

Any issues resolved by technical support will be added to the knowledge base for future reference by customer support personnel.

**7. Functionality for kiosk operators**

The kiosks will be operated at heavy footfall locations like airports, railway stations, shopping malls and markets. The kiosk operator accounts will be created by system administrator. The usual standard login, logout and password update functionality will be provided to the kiosk operator.

**7.1 Ride booking (with optional enroute food delivery) functionality**

Ride booking and optional enroute food delivery will be performed by the operator on behalf of the passenger at kiosk, based on passenger inputs, as described in sections 1.3 and 1.4.

For such bookings, the passenger will need to pay 10% ride payment in advance.

**All-En-Route Software Architecture**

**1. All-En-Route software overview**

All-En-Route software will logically follow layered architecture supporting a scale of millions of users using the All-En-Route user applications in the presentation layer, served by multi-site geographically replicated business and persistence layers (core services and data store), supporting all the required quality attributes listed in the “All-En-Route Functional Architecture and Architecture Principles” document (referred to as “Functional Architecture” from this point onward).

|  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| **Presentation Layer**  [All-En-Route user applications] | Rider application | | | | Driver application | | | | | Operations application | | |
| **Business Layer**  [All-En-Route core business logic] | User management service | Mapping service | | Ride management service | | | Payment processing service | | Food-delivery processing service | | | Analytics service |
| **Persistence Layer** | Data store service | | | | | | | | | | | |
| User and Cab data store | | Ride data store | | | Payment data store | | Analytics data store | | | Operations data store | |

**Fig 1: All-En-Route high level architecture layers**

The presentation layer will consist of user applications for passenger (rider), cab-provider (driver) and operations. The business layer will consist of core services for business logic and persistence layer will consist of data store (management) service for all the data stores. The User and Cab data store will be contextually referred to as User data store or Cab data store in the document. The data stores will be replicated to designated centralized site locations for backup and disaster recovery reasons. Analytics data stores will also be operated only in such central locations to allow for full data access for the analytics across all sites. The Operations data store will have customer issues and knowledge base data. The knowledge base data will also be replicated and merged across all sites for optimal issue resolutions.

**2. All-En-Route user applications**

All-En-Route will have separate **applications for riders** (used by passengers and kiosk operators), **drivers** (used by cab-providers)and **operations** (used by business owner, system administrator, customer and technical support personnel). The applications will mainly constitute **presentation layer** and **minimum necessary business rules processing** with **no data or operational state** being **persisted locally**.

**2.1 Rider Application**

The functionality available in this application will be controlled by role-based access control (**RBAC**). The **roles** will include **passenger** and **kiosk operator**. It will be available as **mobile application for passengers** and **web application for kiosk operators**.

It will provide **client-side interface** for **account management, ride management, location tracking, food-delivery order and payments**.

**2.2 Driver Application**

This will be available as **mobile application for drivers (cab providers)**. It will provide **client-side interface** for **onboarding, authentication, ride management, location tracking and payments**.

**2.3 Operations Application**

The functionality available in this application will be controlled by role-based access control (**RBAC**). The **roles** will include **system administrator, customer support personnel, technical support personnel** and **business owner.**

For **system administrator** role, it will be available as **web application** to provide **interface for user management, service and platform configuration, install, uninstall and upgrades**.

For **customer support personnel** and **technical** **support personnel** roles, it will be available as **web application** to provide **interface for customer issues management and knowledge base management operations**.

For **business owner** role, this will be available as **web application** that runs on laptop **as well as mobile application** that runs on mobile device. It will provide **interface for analytics functionality** for business owner.

**2.4 Application level forward traceability**

|  |  |
| --- | --- |
| **Functionality**  **[Ref: Sections 1 to 7 in Functional Architecture document]** | **All-En-Route Application**  **[Ref: Sections 3 to 5 in this document]** |
| 1. Functionality for passengers | 3. Rider Application |
| 7. Functionality for kiosk operators |
| 2. Functionality for cab providers | 4. Driver Application |
| 3. Functionality for cab aggregator (business owner) | 5. Operations Application |
| 4. Functionality for system administrator |
| 5. Functionality for customer support personnel |
| 6. Functionality for technical support personnel |

**3. Rider Application**

[Note: Specific core service and entry point invocation details are provided in Section 7 of this document]

**3.1 Application function level forward traceability**

|  |  |
| --- | --- |
| **Functionality**  **[Ref: Sections 1 and 7 in Functional Architecture document]** | **All-En-Route Rider Application**  **[Ref: Section 3 in this document]** |
| 1. Functionality for passengers | 3. Rider Application |
| 1.1 Account management | 3.2.1 Account management interface |
| 1.1.1 Registration | 3.2.1.1 Signup |
| 1.1.2 Authentication | 3.2.1.2 Login and logout |
| 1.1.3 Profile Management | 3.2.1.3 Profile |
| 1.2 Mapping services | 3.2.2 Map interface |
| 1.2.1 Locations and routes | 3.2.2.1 Map interface (Source, destination and route) |
| 1.2.2 Location tracking | 3.2.2.2 Map interface (Periodic cab location update) |
| 1.3 Ride management (and 7.1 Ride booking) | 3.2.3 Ride management interface |
| 1.3.1 Ride request creation | 3.2.3.1 Ride request creation and assignment |
| 1.3.2 Ride assignment and notification | 3.2.3.2 Cab assignment notification |
| 1.3.3 Ride cancellation | 3.2.3.3 Ride cancellation |
| 1.4 Enroute food delivery | 3.2.4 Food delivery interface |
| 1.4.1 Food stops and options listing for ride route | 3.2.4.1 Food stops and options listing for ride |
| 1.4.2 Food order delivery functionality for ongoing ride | 3.2.4.2 Food order for ride |
| 1.5 Payments management | 3.2.5 Payments interface |
| 1.5.1 Ride payments | 3.2.5.1 Ride payment |
| 1.5.2 Food delivery order payments | 3.2.5.1 Ride payment (includes food order payments) |

**3.2 Application modules and functional flows**

The application will have following **modules**:

* Account management interface
* Map interface
* Ride management interface
* Food delivery interface
* Payment interface

**3.2.1 Account management interface**

This module will provide interface for **registration, authentication and profile management** functionality described in “Section 1.1. of Functional Architecture” document.

**3.2.1.1 Signup**

Signup or self-registration will be available to passengers only, kiosk operator accounts will be predefined by system administrator. The passenger will use registration UI screen to enter registration details. **Mandatory input** checks and **standard input format validations** will be performed on the information entered by passenger. Personally Identifiable Information (**PII**) will be **encrypted** for privacy reasons and **password** will be **one-way-hashed** for security.

It will invoke core service entry points to verify passenger details and then to persist verified passenger details. On success, user will be registered user and can login and start using the application. On failure, user will be provided with appropriate error messages and allowed to attempt registration again.

**3.2.1.2 Login and logout**

The passenger or (kiosk) operator will use login UI screen to log into the application. **Role (passenger or kiosk operator), UserID and password inputs** will be provided by user on login screen. The **password** will be **one-way-hashed**. The user entered credentials will be verified against persisted credentials in the User data store using core service entry point. On successful verification, user will be logged into the application and can start using the application. On failure, user will be provided with appropriate error messages and allowed to attempt login again with a maximum retry limit of three. Logged in user will have “Logout” option available in application UI to log out of the application.

**3.2.1.3 Profile**

Once successfully logged in, user can choose to update the profile information. Once user confirms updates to the profile, **PII** information in the profile details will be **encrypted**. The user entered profile updates will be persisted in the User data store using core service entry point and status will be communicated to the user.

**3.2.2 Map interface**

This module will provide interface for **location, location tracking and ride routes** functionalities as described in “Section 1.2 of Functional Architecture” document.

**3.2.2.1 Map interface (Source, Destination and Route)**

The user will be able to see the **map view** and provide source (pickup) and destination (drop-off) **locations**. The recent (upto 10) location searches will be displayed. The selected locations will be displayed as markers on the map. Once user confirms the locations, the **route** information will be displayed on the map view.

**3.2.2.2 Map interface (Periodic cab location update)**

Once the ride is confirmed, the user will be able to see **live location of the assigned cab** on the map view. The live location will be available throughout the ride. The user will also be able to **share** the **live location** for user safety purposes.

**3.2.3 Ride management interface**

This module will provide interface for **ride request, cab assignment notification and ride cancellation** functionalities as described in “Section 1.3 of Functional Architecture” document.

**3.2.3.1 Ride request creation and assignment**

The user will be able to view the **ride options** - all the available cab types and price for the type of cab for the given route. The user can select one cab type and proceed with setting ride time. Ride time can be set to now or schedule for later. The **user selected ride option** will be **persisted** in the Ride data storeas ride request using core service entry point.

**3.2.3.2 Cab assignment notification**

The user will be **presented with status of cab assignment**. The interface will get push notification or invoke core service entry point to get assigned cab and driver details periodically. **Once** the ride is **assigned**, it will present the ride information received (**ride OTP, details of cab and driver) to the passenger**. The ride will be maintained in the **upcoming rides** for the passenger before start of the ride. The ride assignment changes (regarding the assigned cab and driver, if any) will be push notified or queried periodically using core service entry point and presented to the user.

**3.2.3.3 Ride cancellation**

User will be allowed to **cancel** the **upcoming ride** request before the ride assignment by clicking on the “Cancel Ride” option available on the UI screen. The cancellation will result in invocation of core service entry point to handle cancellation processing according to the logic described in “Section 1.3.3 of Functional Architecture” document.

**3.2.4 Food delivery interface**

This functionality will be available **only in case of long-distance rides**. It will provide **food stops and food options listing** for the upcoming or ongoing ride and **food order functionality** as described in “Section 1.4 of Functional Architecture” document.

**3.2.4.1 Food stops and options listing for ride**

It will invoke the core service entry point to get **list of all the available food stops on current ride route** as per the logic described in “Section 1.4.1 of Functional Architecture” document. User will be able to see the food stops list and once the user selects a food stop, it will provide the list of available food options using core service entry point.

**3.2.4.2 Food order for ride**

The user will be able to select from **food options presented for selected food stop**. Once selected and confirmed, it will invoke core service entry point to process and persist the food order and present the status to the user.

**3.2.5 Payments interface**

This module will provide interface for **ride (and food bill, if any) payment functionality** as described in “Section 1.5 of Functional Architecture” document.

**3.2.5.1 Ride payment**

Once the ride is marked finished by driver, the ride payment amount will be fetched using core service entrypoint. The ride payment interface will allow user to make the ride payment. User will be provided with an interface with **digital payment options**: Internet banking, Debit/Credit card and UPI. After selection of the payment method, interface for payment authentication will be presented. The **payment details entered** by user will be **masked** when displayed on the interface and will **not be stored for security reasons**. The **payment transaction details** will be **encrypted** before invoking core service to **process and persist** payment transaction in Payment data store and payment status response will be presented to the user.

**3.3 Application architecture**

**[Presentation Layer]**

**Rider application**

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| **UI and minimum necessary business logic** | | | | | | |
| Account management UI | Map UI | Ride management UI | Food delivery UI | Payments UI | Encyption | Hashing |
| Inputs processing and core service entry points invocation | Inputs processing and core service entry points invocation | Inputs processing and core service entry points invocation | Inputs processing and core service entry points invocation | Inputs processing and core service entry points invocation |

**REST APIs**

**[Business Layer (Core Services)]**

|  |
| --- |
| **Firewall** [Service discovery and security] |
| **Load Balancer** |
| **All-En-Route core business services** |

**Data store service APIs**

User and Cab Data Store

Payment Data Store

Ride Data Store

**[Persistence Layer ]**

**Fig 2: Architecture for Rider Application**

Rider application will be available as mobile application for passengers and web application for kiosk operators. It will primarily provide user interface for the functionality described in section 3.2 of this document and minimum necessary business logic for input validations, encryption, one-way-hashing and invoking appropriate core service entry points of All-En-Route.

The mobile application will be available for both iOS and Android OS and will run in native runtime environment available in respective OS. The web application will run in browser (Chrome, Internet Explorer and Safari) on laptop.

**4. Driver Application**

[Note: Specific core service and entry point invocation details are provided in Section 7 of this document]

**4.1 Application function level forward traceability**

|  |  |
| --- | --- |
| **Functionality**  **[Ref: Section 2 in Functional Architecture document]** | **All-En-Route Driver Application**  **[Ref: Section 4 in this document]** |
| 2. Functionality for Cab providers | 4. Driver Application |
| 2.1 Account management | 4.2.1 Account management interface |
| 2.1.1 Onboarding | 4.2.1.1 Onboarding |
| 2.1.2 Login and logout | 4.2.1.2 Login and logout |
| 2.1.3 Profile Management | 4.2.1.3 Profile |
| 2.2 Mapping services | 4.2.2 Map interface |
| 2.3 Ride management | 4.2.3 Ride management interface |
| 2.3.1 Ride request assignment | 4.2.3.1 Ride assignment |
| 2.3.2 Ride processing | 4.2.3.2 Ride processing |
| 2.3.3 Ride cancellation | 4.2.3.3 Ride cancellation |
| 2.4 Payments management | 4.2.4 Payments interface |

**4.2 Application modules and functional flows**

The application will have following **modules**:

* Account management interface
* Map interface
* Ride management interface
* Payment interface

**4.2.1 Account management interface**

This module will provide interface for **onboarding, authentication and profile management** functionality described in “Section 2.1. of Functional Architecture” document.

**4.2.1.1 Onboarding**

Onboarding interface option will be available for driver (cab provider) on application startup. As described in “Section 2.1.1 of Functional Architecture” document, the driver will use registration UI screen to enter registration details and upload documents along with cab details. Documents will have to be uploaded in form of **JPG** or **PDF**.

**Mandatory input** checks and **standard input format validations** will be performed on the information entered by driver. Personally Identifiable Information (**PII**) will be **encrypted** for privacy reasons and **password** will be **one-way-hashed** for security. The interface will then invoke core service entry point to persist the driver and cab details, upload documents to User and Cab data store and start the verification process. Phone number and e-mail will be verified using OTP and e-mail verification link. Uploaded documents will be verified offline for authenticity within **2-3 working days**. On successful verification, cab drivers will be notified via phone and e-mail and they will need to **agree on a contract specifying all the terms and conditions**. On successful verification and contract signing, cab and driver details persisted will be marked active and driver will be notified of the same.

**4.2.1.2 Login and logout**

Successfully onboarded cab providers will use login UI screen to log into the application. UserID and password inputs will be provided by user on login screen. The **password** will be **one-way-hashed**. The user entered credentials will be verified against persisted credentials in the User data store using core service entry point. On successful verification, user will be logged into the application and can start using the application. On failure, user will be provided with appropriate error messages and allowed to attempt login again with maximum retry limit of three. Logged in user will have “Logout” option available in application UI to log out of the application.

**4.2.1.3 Profile**

Once successfully logged in, user can choose to update the profile information. Once user confirms updates to the profile, **PII** information in the profile details will be **encrypted**. The user entered profile updates will be persisted in the User data store using core service entry point and status will be communicated to the user.

**4.2.2 Map interface**

This module will provide interface for viewing assigned ride source and destination along with driving route and directions as described in “Section 2.2 of Functional Architecture” document.

The user will see a map view on the screen **showing the source and destination locations of the ride**, the **route to take** and the estimated **time** ofarrival (ETA) of destination. **On ride start, driving directions** will be provided in real-time. The information to populate the UI will be obtained using core service entry points.

**4.2.3 Ride management interface**

This module will provide interface for **ride assignment, ride processing and ride cancellation** functionality as described in “Section 2.3 of Functional Architecture” document.

**4.2.3.1 Ride assignment**

The **cab providers will see their matched rides in the UI.** The matching will take place in core services as per the business rules and requirements, as described in the “Section 2.3.1 of Functional Architecture” document. The cab provider will be able to **select and accept** one from available ride requests. The service platform will then assign the ride request to the cab provider using criteria as described in the “Section 2.3.1 of Functional Architecture” document. The cab provider will be then notified of ride assignment. The cab will be marked busy from the beginning of the ride until 5 minutes before the end time.

The matched ride data will be obtained by invoking core service entry point. Once a ride is accepted by driver, ride request will be updated in Ride data store.

**4.2.3.2 Ride processing**

To start a ride, the driver will get a UI screen to enter an OTP provided by the passenger. The OTP data will be validated by invoking core service entry point. On successful OTP validation, the ride will begin. Cab location tracking (source, destination, route, current location and ETA) will be available to the rider and the driver till the ride ends. On reaching the destination, the driver will use an option in the UI to finish the ride, after which the payment screen will be available to passenger.

**4.2.3.3 Ride cancellation**

The driver will have an option to cancel accepted and assigned ride as per the rules described in “Section 2.3.3 of Functional Architecture” document. The ride cancellation will be updated in ride request status in the Ride data store by invoking core service entry point.

**4.2.4 Payments interface**

The payments for driver (cab providers) will be settled on daily basis using daily batch jobs in the system. This module will provide interface for **ride payment** functionality as described in “Section 2.4 of Functional Architecture” document. Payments to drivers (cab providers) will be **settled** by the service platform **on a daily basis**. The drivers (cab providers) will receive consolidated statement for the rides completed and penalties, if any, for the day, on their registered e-mail. The same can also be seen in the interface of the application. The payment records will be **encrypted** while in-transit and at rest and will only be accessible on entering the password. The payment records will be obtained from the Payment data store by invoking core service entry point.

**4.3 Application architecture**

**[Presentation Layer]**

**Driver application**

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **UI and minimum necessary business logic** | | | | | |
| Account management UI | Map UI | Ride management UI | Payments UI | Encyption | Hashing |
| Inputs processing and core service entry points invocation | Inputs processing and core service entry points invocation | Inputs processing and core service entry points invocation | Inputs processing and core service entry points invocation |

**REST APIs**

**[Business Layer (Core Services)]**

|  |
| --- |
| **Firewall** [Service discovery and security] |
| **Load Balancer** |
| **All-En-Route core business services** |

**Data store service APIs**

User and Cab Data Store

Payment Data Store

Ride Data Store

**[Persistence Layer]**

**Fig 3: Architecture for Driver Application**

Driver application will be available as mobile application for drivers. It will primarily provide user interface for the functionality described in section 4.2 of this document and minimum necessary business logic for input validations, encryption, one-way-hashing and invoking appropriate core service entry points of All-En-Route.

The mobile application will be available for both iOS and Android OS and will run in native runtime environment available in respective OS.

**5. Operations Application**

[Note: Specific core service and entry point invocation details are provided in Section 7 of this document]

**5.1 Application function level forward traceability**

|  |  |
| --- | --- |
| **Functionality**  **[Ref: Sections 3, 4, 5 and 6 in Functional architecture document]** | **All-En-Route Operations Application**  **[Ref: Section 5 in this document]** |
| 4. Functionality for system administrator | 5. Operation Application |
| 3,4,5,6 Standard login and logout | 5.2.1 Authentication interface |
| 4.1 User account management | 5.2.2 Account management interface |
| 4.2 Service platform installation management | 5.2.3 Service platform management interface |
| 5. Functionality for customer support personnel | 5. Operation Application |
| 5.1 Customer issue processing | 5.2.4 Customer issue interface |
| 5.2 Support knowledge base search | 5.2.5 Knowledge base interface |
| 6. Functionality for technical support personnel | 5. Operation Application |
| 6.1 Customer issue processing | 5.2.6 Technical issue interface |
| 6.2 Support knowledge base update | 5.2.7 Knowledge base management interface |
| 3. Functionality for cab aggregator (business owner) | 5. Operation Application |
| 3.1 Basic charts | 5.2.8 Basic charts interface |
| 3.2 Analytics charts | 5.2.9 Analytics charts interface |
| 3.3 Contracts and compliance information | 5.2.10 Dashboard interface |

**5.2 Application modules and functional flows**

The application will have following **modules:**

* Authentication interface
* Account management interface
* Service platform management interface
* Customer issue interface
* Knowledge base interface
* Technical issue interface
* Knowledge base management interface
* Basic charts interface
* Analytics charts interface
* Dashboard interface

**5.2.1 Authentication interface**

This module will provide **user authentication** and **authorization** functionality using role-based access control **(RBAC)**. The **roles** - System administrator, Customer support personnel, technical support personnel and Business owner will be **preassigned** to the users.

User will need to change the password on first login. User then will enter **user ID** and **password** to log into the application. For **privacy** reasons, the password will be **masked** while being entered. It will then **invoke core service entry point to verify the credentials (with password one-way-hashed for security)** and get the preassigned role. On successful verification, user will be logged into the application with preassigned role with RBAC enforced, permitting only the functionalities authorized for the role. On failure, user will be provided with appropriate error messages and allowed to attempt login again with maximum retry limit of three. Logged in user will have “Logout” option available in application UI to log out of the application.

**5.2.2 Account management interface**

The system administrator and business owner accounts will be preconfigured for the system. The customer support personnel, technical support personnel and kiosk operator accounts will be created by the system administrator as needed.

This module will be only available to **system administrator**. It will provide the administrator with an UI interface to **Create**, **View**, **Update** and **Delete** (CRUD) customer support personnel, technical support personnel and kiosk operator accounts for the service platform. While updating any account, Personally Identifiable Information (**PII**) will be **encrypted** for privacy reasons and **password** will be **one-way-hashed** for security. Administrator can also **block** the cab providers who fail to comply with automated periodic regulatory checks. The account registrations and profile updates will be persisted in the User data store using core service entry point.

**5.2.3 Service platform management interface**

This module will allow **system administrator** to **install, configure, upgrade, uninstall All-En-Route software** on the initial hardware and **apply security patches** for the service platform. It will provide separate UI for **service platform health and resource capacity monitoring**. Administrator will also be able to **manage and configure batch jobs for** driver (cab provider) and food-delivery service partner daily **payment settlements** and automated periodic regulatory **compliance checks** of the drivers and cabs. The install, configure, upgrade, uninstall and patch processes will use the native mechanism available in the deployment environment. The batch jobs will use core service entry points for payments processing. The compliance checks will start offline verification process for verifications due as per regulations. The verification process statuses will be uploaded to the system and batch jobs will produce list of drivers and/or cabs that will need to be deactivated due to non-compliance.

**5.2.4 Customer issue interface**

This module is for **customer support personnel**. It will show the customer raised issues sorted with severity (high, medium or low) level as mentioned by the issuer. The personnel will be able to **view** the assigned issues and **resolve** or **forward** it to the technical support. If the issue is resolved, customer will be notified accordingly. The updates will be persisted in the Operations data store using the core service entry point.

**5.2.5 Knowledge base interface**

This module is for **customer support personnel**. It will provide an UI based interface to **search** for the customer issue from the Knowledge Base (KB). If the resolution is present in KB, the personnel will apply the same solution and **resolve** the issue or else forward it to technical support. It will invoke the core service entry point to search the knowledge base in the Operations data store.

**5.2.6 Technical issue interface**

This module is for **technical support personnel**. It will show the issues forwarded by the customer support in severity order. The personnel will be able to **resolve** the issue and update the issue status. The updates will be persisted in the Operations data store using the core service entry point.

**5.2.7 Knowledge base management interface**

This module is for **technical support personnel**. It invokes the core service entry point to manage the Knowledge Base. The personnel can add, remove or update the resolved technical issue to the knowledge base for future reference by the customer support.

**5.2.8 Basic charts interface**

This module is for the **business owner**. The basic charts functionality will present user with predefined charts and allow user to select the dimensions, metrics and time granularity from prepopulated lists for the chart. It will invoke core service entry point to obtain the chart data for selected attributes from Analytics data store.

**5.2.9 Analytics charts interface**

This module is for the **business owner**. This will allow the user to **configure comparison charts** and **trends charts and** allow user to **explore basic charts on second dimension**. It will invoke the core service entry point to persist the configured chart definitions and obtain the chart data from Analytics data store. The user will be allowed to delete any charts.

**5.2.10 Dashboard interface**

This module is for the **business owner**. The dashboard will provide information regarding regulatory compliance statuses, business licenses, partner service subscription or contract fees and renewal due dates.

**5.3 Application Architecture**

**[Presentation Layer]**

**Operations application**

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| **UI and minimum necessary business logic** | | | | | | | |
| Account management UI | Service platform management UI | Customer and technical support UI | Knowledge base management UI | Dashboard UI | Charts UI | Encryption | Hashing |
| Inputs processing and core service entry points invocation | Inputs processing and core service entry points invocation | Inputs processing and core service entry points invocation | Inputs processing and core service entry points invocation | Inputs processing and core service entry points invocation | Inputs processing and core service entry points invocation |

**System administrator Support Business owner**

**REST APIs**

**[Business Layer (Core Services)]**

|  |
| --- |
| **Firewall** [Service discovery and security] |
| **Load Balancer** |
| **All-En-Route core business services** |

**Data store service APIs**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Data store service | | | | |
| User and Cab data store | Ride data store | Payment data store | Analytics data store | Operations data store |

**[Persistence Layer]**

**Fig 4: Architecture for Operations Application**

Operations application will be available as web application for system administrator, customer support personnel, technical support personnel and as web application as well as mobile application for business owner. It will primarily provide user interface for the functionality described in section 5.2 of this document and minimum necessary business logic for input validations, encryption, one-way-hashing and invoking appropriate core service entry points of All-En-Route. The mobile application will be available for both iOS and Android OS and will run in native runtime environment available in respective OS. The web application will run in browser (Chrome, Internet Explorer and Safari) on laptop.

**6. All-En-Route core and platform services**

Based on characteristics of various architectural styles for loosely coupled highly cohesive modularization with anytime anywhere availability, responsiveness, deployability and maintainability requirements, **All-En-Route core business logic will use service-oriented architecture** and will be manifested as microservices. All the service entry points will use common service entry point for logging.

**All-En-Route** business logic will be available in following **core services**:

* UserManagement
* MappingService
* RideManagement
* PaymentProcessing
* Food-deliveryProcessing
* AnalyticsService

Following are **common services** used by core services:

* DataStoreService
* ValidationService
* AuditService

**6.1 User management service (UserManagement)**

This service will provide user account management functionality for registration, authentication, and profile management.

It will have following entry points:

* GenerateOTP
* ValidateOTP
* GenerateEmailVerificationLink
* ValidateEmailVerificationLink
* RegisterUser
* AuthenticateUser
* UpdateProfile

**6.1.1 GenerateOTP**

For given user ID and phone number inputs, it will generate OTP for given user ID using ValidationService:GenerateOTP and send it to user phone number using ValidationService:SendOTP. The OTP will be stored in User data store marked for deletion after its expiry.

**6.1.2 ValidateOTP**

For given user ID and phone number inputs and OTP, it will validate OTP using ValidationService:ValidateOTP for match and expiry. It will remove OTP from User data store once processed, using ValidationService:DeleteOTP.

**6.1.3 GenerateEmailVerificationLink**

For given user ID and e-mail address inputs, using ValidationService:GenerateEmailVerificationLink it will generate e-mail verification link for given user ID and send the link to user e-mail address using ValidationService:SendEmailVerificationLink. The e-mail verification link will be stored in User data store marked for deletion after its expiry.

**6.1.4 ValidateEmailVerificationLink**

For given user ID and e-mail address inputs and e-mail verification link, using ValidationService:ValidateEmailLink it will validate e-mail verification link for match and expiry. Once processed, it will remove e-mail verification link using ValidationService:DeleteEmailLink from User data store.

**6.1.5 RegisterUser**

It will persist input user registration details in User data store using DataStoreService:CreateOrUpdateDataRecord.

**6.1.6 AuthenticateUser**

It will validate input user credentials against user credentials available in User data store using VerificationService:ValidateCredentials, it will return the status and preassigned role of the validated user.

**6.1.7 UpdateProfile**

It will persist input profile registration details in User data store using DataStoreService:CreateOrUpdateDataRecord.

**6.2 Mapping service (MappingService)**

This service will invoke predetermined external geocoding and routing service provider interfaces to get the required functionality. Using the external mapping service provider interfaces, it will provide for **location latitude and longitude (lat-long), source to destination routes and cab location tracking** functionality. It will have following **entrypoints**:

* GetMapViewForLocation
* GetMapViewForRoute
* StartMapViewUpdatesForCabLocation
* StopMapViewUpdatesForCabLocation

**6.2.1 GetMapViewForLocation**

For given textual location input, GetMapViewForLocation will **call the predetermined external geocoding and mapping service provider interface to get** the **lat-long and map view information** for the (Rider and Driver) application to render the map view with location marker.

[The external geocoding service provider to use will be predetermined after evaluating options (such as Google Maps Geocoding API or TomTom Search API) during design phase.]

**6.2.2 GetMapViewForRoute**

For given source and destination location inputs, GetMapViewForRoute will **call the predetermined external routing and mapping service provider interface to get the routing and map view information** for the (Rider and Driver) application to render the map view with route.

[The external routing service provider to use will be predetermined after evaluating options (such as Google Maps Directions API or TomTom Routing API) during design phase.]

**6.2.3 StartMapViewUpdatesForCabLocation (Cab Location Tracking)**

For given ride ID input for current ride**, long polling** will be setup for Rider and Driver applications **to receive initial map view information with source, destination, route for the ride and continuous map view updates for cab location** based on driver’s smartphone GPS coordinate updates received periodically.

**6.2.4 StopMapViewUpdatesForCabLocation (Cab Location Tracking)**

**On current ride finish** for given cab ID, **long polling setup** for Rider and Driver applications to receive map view updates for cab location **will be taken down**.

**6.3 Ride management service**

This service will provide for **route, tariff and cab availability for given ride, scheduling, assignment and cancellation of rides** functionality. It will have following **entry points**:

* GetRideOptions
* CreateAndAssignRide
* GetAssignedCabDetails
* AssignRide
* StartRide
* EndRide
* CancelRide

**6.3.1 GetRideOptions**

This entrypoint provides **map view with source, destination locations, route, distance, ETA, tariff (by cab types) and cabs available in the vicinity of source location** for ride information. For given source and destination inputs, GetRideOptions will call **GetMapViewForLocation and GetMapViewForRoute entrypoints of the MapService** to **get map view with location lat-longs** and **route** information. It will **use** the **route distance and ETA information to compute tariffs for various cab types**. Based on the continuous **cab’s location** tracking (GPS Coordinates) **and Geospatial Indexing**, cab locations in the vicinity will be obtained **for reporting** on map view interface.

**6.3.2 CreateAndAssignRide**

**For given ride** (Now), CreateRide will **create ride request**, **persist** it in ride data store and will **assign** using the logic described in “Section 2.3.1.1 of Functional Architecture” document for regular ride assignment and “Section 2.3.2.2 of Functional Architecture” document for long-distance ride assignment. It will return the ride ID and status of processing once the ride request is created and ride assignment will continue asynchronously. The status of assigned ride will be updated in Ride data store once done.

**For given ride with schedule time** (!= Now), CreateRide will **create ride request**, **persist** it in ride data store. Ride request will be associated **with timer** that will go off at [ride start time – predefined buffer]. The predefined buffer will be 30 minutes for regular rides and 2 hours for long distance rides. When the timer goes off, **ride assignment** will **begin**. It will return the ride ID and status of processing once the ride request is created and ride assignment will continue asynchronously. The status of assigned ride will be updated in Ride data store once done.

**6.3.3 GetAssignedCabDetails**

When invoked with a ride ID, it will return **current status of ride assignment and cab details, driver details and ride OTP if assignment is done** from Ride data store.

**6.3.4 AssignRide**

For given ride ID and driver ID inputs, AssignRide will update the status of the ride to **ASSIGNED** in the Ride data store and ride request will be updated with driver ID and cab ID.

**6.3.5 StartRide**

For given ride ID marked started with OTP, StartRide will match the OTP in the persisted ride request and update the status of the ride request to **STARTED** along with start time in the Ride data store.

**6.3.6 EndRide**

For given ride ID marked finished by user, EndRide will update the status of the ride to **FINISHED** along with end time in the Ride data store.

**6.3.7 CancelRide**

For given ride ID marked cancelled by user, CancelRide will update the status of the ride to **CANCELLED** along with cancellation time in the Ride data store.

**6.4 Payment processing service**

This service will process rides (food-delivery and penalties) payments for rides marked FINISHED or CANCELLED, daily driver (cab provider) payments and daily food-delivery service partner payments. It will call payment gateway service interfaces. It will have following **entry points**:

* GetRideDues
* ProcessPassengerPayment
* ProcessDriverPayment
* ProcessFoodDeliveryPartnerPayments

**6.4.1 GetRideDues**

When invoked with ride ID, GetRideDues will compute the ride payment dues based on ride tariff, food bills of the ride, cancellation charges for current ride if applicable (based on cancellation status and start and cancellation time of the ride) and passenger penalties pending if any.

**6.4.2 ProcessPassengerPayment**

When invoked with payment method type and payment method details, it will call payment gateway service interface for given payment method with given payment method details. The payment transactions will be stored in Payment data store.

**6.4.3 ProcessDriverPayment**

When invoked with list of drivers (cab provider) payment transactions for the day, it will fetch payment method type and details for each driver for each transaction. It will then call payment gateway service interface for the payment method with payment method details to make the payments. All the payment transactions will be stored in Payment data store.

**6.4.4 ProcessFoodDeliveryPartnerPayments**

When invoked with list of food-delivery service partner(s) payment transactions for the day, it will fetch payment method type and details for the food-delivery service partner for each transaction. It will then call payment gateway service interface for the payment method with payment method details. All the payment transactions will be stored in Payment data store.

**6.5 Food-delivery processing service**

This service will provide food stops listing, food menu listing, food-deliver ordering and status interface using respective interfaces of food-delivery partner service and present that information in the Rider application UI. It will have following **entry points**:

* ListFoodStops
* ListFoodOptions
* OrderFoodDelivery

**6.5.1 ListFoodStops**

When invoked with route details, it will compute food stops available on given route as per the logic described in “Section 1.4.1 in Functional Architecture” document. It will persist the food stops in the ride request in Ride data store.

**6.5.2 ListFoodOptions**

When invoked with food stop input, it will query the food-delivery partner service(s) for food options available for given food stop.

**6.5.3 OrderFoodDelivery**

When invoked with food stop and food options selected for ordering inputs, it will place the food delivery order for selected food options using food-delivery partner service(s) interfaces and persist the food-delivery order in ride request in Ride data store.

**6.6 Analytics service**

This service will provide functionality for predefined and configurable charts, trends, comparisons insights for business owners. It will have following **entry points**:

* GetCharts
* ConfigureChart
* DeleteChart

**6.6.1 GetCharts**

When invoked, it will **return currently predefined basic and preconfigured analytics charts data** from Analytics data store.

**6.6.2 ConfigureChart**

When invoked with **type of chart, dimensions, metrics and time granularity**, it will **generate analytics chart query that will run as per given time granularity** and **persist** it in Analytics data store **as preconfigured chart**.

**6.6.3 DeleteChart**

When invoked with chart ID, it will **delete** that **predefined or preconfigured chart definition** from Analytics Data Store.

**6.7 Data store service**

This platform service will provide data create, read, update and delete (CRUD) operations to all the core services for specified data stores. It will use standard opensource NoSQL database that supports required data models. The NoSQL database to use will be predetermined after evaluating options (such as Cassandra or MongoDB) during design phase. It will have following **entry points**:

* CreateOrUpdateDataRecord
* GetDataRecord
* DeleteDataRecord

**6.7.1 CreateOrUpdateDataRecord**

When invoked with data record and data store, it will create or update the data record in the given data store and return the status.

**6.7.2 GetDataRecord**

When invoked with query attributes and data store, it will retrieve the data record based on given query attributes from the given data store and return the status and data record if found.

**6.7.2 DeleteDataRecord**

When invoked with attributes and data store, it will delete the data record matching the given attributes in the given data store and return the status.

**6.8 ValidationService**

This common service will provide OTP generation, validation, e-mail link generation and validation functions. It will have following **entry points:**

* GenerateOTP
* SendOTP
* ValidateOTP
* DeleteOTP
* GenarateEmailVerificationLink
* SendEmailVerificationLink
* ValidateEmailVerificationLink
* DeleteEmailVerificationLink

**6.8.1 GenerateOTP**

GenerateOTP function will generate an OTP for given user ID or ride ID.

**6.8.2 SendOTP**

SendOTP will SMS the OTP to the user associated with the user ID or ride ID.

**6.8.3 ValidateOTP**

ValidateOTP will validate OTP for given user ID or ride ID against the persisted OTP in user record in User data store or ride request in Ride data store.

**6.8.4 DeleteOTP**

DeleteOTP will delete the OTP after processing or expiry.

**6.8.5 GenerateEmailVerificationLink**

GenerateEmailVerificationLink function will generate an EmailVerificationLink for given user ID.

**6.8.6 SendEmailVerificationLink**

SendEmailVerificationLink will email the EmailVerificationLink to the user associated with the user ID.

**6.8.7 ValidateEmailVerificationLink**

ValidateEmailVerificationLink will validate EmailVerificationLink for given user ID against the persisted EmailVerificationLink in user record in User data store.

**6.8.8 DeleteEmailVerificationLink**

DeleteEmailVerificationLink will delete the EmailVerificationLink after processing or expiry.

**6.9 AuditService**

This common service will provide WriteLog entrypoint to write log records to Operations data store.

**7. Integration Architecture**

**7.1 Internal and External integrations**

Internal integration refers to integration across multiple applications of the system and external integration refers to integration with external systems.

Internally, the All-En-Route user applications will interact with the All-En-Route core services using REST API entry points. The services will persist and retrieve data using data store services.

External services used by All-En-Route will include external mapping services, external food-delivery partner services and payment gateway services. All-En-Route core services will invoke the interfaces provided by these external services.

Table below depicts the integrations among the applications and services in the All-En-Route system. Invocations by user applications will be inbound to the invoked service which will provide the respective entry point. Invocations of external service (payment gateway, mapping and food delivery partner services) entry points by All-En-Route services will be outbound and will be provided by external services.

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **ID** | **Invoking Application** | **Invoked Service** | **Integration Description** | **Inbound or Outbound** | **Data integration or functional integration** |
| 7.2.1 | Rider Application | UserManagement | Passenger details verification | Inbound | Functional integration |
| 7.2.2 | Rider Application | UserManagement | Passenger registration | Inbound | Functional integration |
| 7.2.3 | Rider Application | UserManagement | Passenger login and logout | Inbound | Functional integration |
| 7.2.4 | Rider Application | UserManagement | Passenger Profile Update | Inbound | Functional integration |
| 7.2.5 | Rider Application | RideManagement | Passenger map views with location and routes | Inbound | Functional integration |
| 7.2.6 | Rider Application | MappingService | Cab location tracking | Inbound | Functional integration |
| 7.2.7 | Rider Application | RideManagement | Ride request creation and assignment | Inbound | Functional integration |
| 7.2.8 | Rider Application | RideManagement | Cab assignment notification | Inbound | Functional integration |
| 7.2.9 | Rider Application | RideManagement | Ride cancellation by passenger | Inbound | Functional integration |
| 7.2.10 | Rider Application | PaymentProcessing | Ride payment | Inbound | Functional integration |
| 7.2.11 | Rider Application | Food-deliveryProcessing | Food stops listing | Inbound | Functional integration |
| 7.2.12 | Rider Application | Food-deliveryProcessing | Food options listing | Inbound | Functional integration |
| 7.2.13 | Rider Application | Food-deliveryProcessing | Food order | Inbound | Functional integration |
| 7.2.14 | Driver Application | UserManagement | Driver details verification | Inbound | Functional integration |
| 7.2.15 | Driver Application | UserManagement | Driver onboarding | Inbound | Functional integration |
| 7.2.16 | Driver Application | UserManagement | Driver login and logout | Inbound | Functional integration |
| 7.2.17 | Driver Application | UserManagement | Driver Profile Update | Inbound | Functional integration |
| 7.2.18 | Driver Application | MappingService | Driver map views with location, routes, location tracking and directions | Inbound | Functional integration |
| 7.2.19 | Driver Application | RideManagement | Ride assignment | Inbound | Functional integration |
| 7.2.20 | Driver Application | RideManagement | Ride processing | Inbound | Functional integration |
| 7.2.21 | Driver Application | RideManagement | Ride cancellation by driver | Inbound | Functional integration |
| 7.2.22 | Driver Application | PaymentProcessing | Daily Payment Settlement | Inbound | Functional integration |
| 7.2.23 | Operations Application | UserManagement | Authentication | Inbound | Functional integration |
| 7.2.24 | Operations Application | UserManagement | Account management | Inbound | Functional integration |
| 7.2.25 | Operations Application | DataStoreService | Customer issues | Inbound | Functional integration |
| 7.2.26 | Operations Application | DataStoreService | Knowledge base search | Inbound | Functional integration |
| 7.2.27 | Operations Application | DataStoreService | Technical issues | Inbound | Functional integration |
| 7.2.28 | Operations Application | DataStoreService | Knowledge base update | Inbound | Functional integration |
| 7.2.29 | Operations Application | AnalyticsService | Basic charts | Inbound | Functional integration |
| 7.2.30 | Operations Application | AnalyticsService | Analytics charts | Inbound | Functional integration |
| 7.2.31 | Operations Application | AnalyticsService | Analytics charts | Inbound | Functional integration |
| 7.2.32 | Operations Application | DataStore | Dashboard for business owner | Inbound | Functional integration |
| 7.2.33 | MappingService | Geocoding, External Mapping service | Location map view | Outbound | Functional integration |
| 7.2.34 | MappingService | Routing, External Mapping service | Route map view | Outbound | Functional integration |
| 7.2.35 | RideManagement | Geospatial Indexing, External Mapping service | Cabs in vicinity and map view | Outbound | Functional integration |
| 7.2.36 | Food-deliveryProcessing | Food-delivery partner service | Food options | Outbound | Functional integration |
| 7.2.37 | Food-deliveryProcessing | Food-delivery partner service | Food order | Outbound | Functional integration |
| 7.2.38 | PaymentProcessing | PaymentGateway | Payment processing | Outbound | Functional integration |
| 7.2.39 | All services | DataStoreService | Data record CRUD | Inbound | Functional integration |
| 7.2.40 | All services | AuditService | Logging | Inbound | Functional integration |
| 7.2.41 | All applications | ValidationService | Phone and e-mail verification | Inbound | Functional integration |

**7.2 Integration details**

**7.2.1 Passenger details verification**

On successful input checks of the information entered by passenger, the **Signup** functionality in **Rider application** will invoke **GenerateOTP entrypoint of UserManagement service** to generate and send the OTP and will invoke **ValidateOTP** **to verify** the **phone number using OTP.** It will invoke **GenerateEmailVerificationLink** **entrypoint of UserManagement service** to generate and send e-mail verification link and will invoke **ValidateEmailVerificationLink to verify** the **e-mail** address **using e-mail verification link**.

**7.2.2 Passenger registration**

On successful verification of user details and user confirmation for registration, the **Signup** functionality in **Rider application** will invoke **RegisterUser entrypoint of UserManagement service** to persist passenger details in User data store. The return status will be communicated to the user.

**7.2.3 Passenger login and logout**

After encrypting user (passenger) credentials, the **AuthenticateUser entrypoint of UserManagement service** will be **invoked** to validate the credentials against the credential’s information available in the User data store and status will be communicated to the user.

**7.2.4 Passenger profile update**

After encrypting PII information in the profile details, the **UpdateProfile entrypoint of UserManagement service** will be invoked to persist the profile updates and status will be communicated to the user.

**7.2.5 Passenger map views with location, routes, ride options**

The map view interface of Rider application will invoke **GetRideOptions entrypoint of RideManagement service** to get the required map view details with source, destination, route, ride options with tariff and ETA information for rendering the map view for the passenger.

**7.2.6 Cab location tracking**

Once the cab is assigned, the map view interface of Rider application will invoke **StartMapViewUpdatesForCabLocation entry point of MappingService service** with the ride ID for cab location tracking using long polling setup to get initial map view information with source, destination, route and continuous map view updates for current cab location. The cab location updates will be available throughout the ride.

**7.2.7 Ride request creation and assignment**

Once the rider selects the ride option and confirms ride, the Rider application will invoke **CreateAndAssignRide entry point of RideManagement service.** The service entry point will return the ride ID and status of the ride creation. The assignment will happen asynchronously and will be push notified or polled for status.

**7.2.8 Cab assignment notification**

The cab assignment screen of Rider application will be refreshed periodically for assigned cab and driver details. The updates are fetched periodically using push notifications from RideManagement service or by calling **GetAssignedCabDetails entry point of RideManagement service.**

**7.2.9 Ride cancellation by passenger**

On cancellation by passenger, Rider application invokes **CancelRide entrypoint of RideManagement service.** The service entrypoint will update the status of the ride request as cancelled in the Ride data store.

**7.2.10 Ride payment**

The payment interface will be activated once the ride is marked finished. The bill information will be either fetched via push notifications or by periodically invoking **GetRideDues entrypoint of PaymentService**. For payment processing, once the payment method is selected, details are entered (masked) and encrypted, Rider application will invoke **ProcessPassengerPayment entry point of PaymentService.** The payment transaction will be processed by the service entry point and persisted in the Payment data store.

**7.2.11 Food stops listing**

Once the ride is confirmed, Rider application will invoke **ListFoodStops entrypoint of Food-deliveryProcessing service** to get the food stops to be presented to the passenger.

**7.2.12 Food options listing**

Once the passenger selects food stop, Rider application will invoke **ListFoodOptions entrypoint of Food-deliveryProcessing service** to get the food options at selected food stop to be presented to the passenger.

**7.2.13 Food order**

Once the passenger selects food from food options to order, the Rider application will call **OrderFoodDelivery entrypoint of Food-deliveryProcessing service**. The service entry point will call outbound interface of food-delivery partner service to place the order and get the confirmation status. It will then respond back with status after updating the ride request with the associated food-order in Ride data store.

**7.2.14 Driver details verification**

On successful input checks of the information entered by driver, the Onboarding functionality in **Driver application** will invoke **GenerateOTP entrypoint of UserManagement service** to generate and send the OTP and will invoke **ValidateOTP** **to verify** the **phone number using OTP.** It will invoke **GenerateEmailVerificationLink** **entrypoint of UserManagement service** to generate and send e-mail verification link and will invoke **ValidateEmailVerificationLink to verify** the **e-mail** address **using e-mail verification link**.

**7.2.15 Driver onboarding**

On successful verification of driver details and start of onboarding process, the **Onboarding** functionality in **Driver application** will invoke **RegisterUser entrypoint of UserManagement service** to persist driver and cab details and documents provided in User data store. The return status will be communicated to the user. After successful offline verification of documents, activation of the driver and cab will be done by system administrator via daily automated batch job of activating verified drivers and cabs and sending SMS and e-mail notifications to the respective drivers (cab providers).

**7.2.16 Driver login and logout**

After encrypting user (driver) credentials, the **AuthenticateUser entry point of UserManagement service** will be **invoked** to validate the credentials against the credential’s information available in the User data store and status will be communicated to the user.

**7.2.17 Driver Profile Update**

After encrypting PII information in the profile details, the **UpdateProfile entry point of UserManagement service** will be invoked to persist the profile updates and status will be communicated to the user.

**7.2.18 Driver map views with location and routes**

Once the driver starts the ride using Driver application UI, it will invoke **StartMapViewUpdatesForCabLocation entry point of MappingService serivce** with the ride ID for cab location tracking using long polling setup to get initial map view information with source, destination, route and continuous map view updates for current cab location. The cab location updates will be available throughout the ride.

**7.2.19 Ride assignment**

The driver will be notified with available matched ride requests list via push notification from RideManagement service. Once the driver accepts one of the rides from the list, Driver application will invoke **AssignRide entrypoint of RideManagement service.** The ride request will be updated by the service entry point for assigned driver and cab in Ride data store.

**7.2.20 Ride processing**

When the driver starts the ride by entering OTP provided by the passenger, the Driver application will invoke **StartRide entrypoint of RideManagement service.** The service entry point will update the status of ride request after matching the OTP provided.

When the driver marks the ride finished using Driver application UI, the Driver application will invoke **EndRide entrypoint of RideManagement service.** The service entry point will update the status of ride request.

**7.2.21 Ride cancellation by driver**

On cancellation by driver, Driver application invokes **CancelRide entrypoint of RideManagement service.** The service entrypoint will update the status of the ride request as cancelled in the Ride data store.

**7.2.22 Daily Payment Settlement**

System will run configured daily automated batch job to settle drivers (cab providers) payments. It will prepare the list of driver (cab providers) payment transactions and call **ProcessDriverPayment entrypoint of PaymentService** with this list. The service entry point will execute the payment transactions to settle the daily driver (cab provider) dues.

**7.2.23 Authentication**

Operations application will invoke **AuthenticateUser entry point of UserManagement service** to validate the credentials against the credential’s information available in the User data store and status and preassigned role of the user will be returned.

**7.2.24 Account management**

For user profiles created by system administrator in Operations application for support staff and kiosk operators, Operations application will invoke **RegisterUser entrypoint of UserManagement service** to register and persist user details in User data store. The return status will be communicated back.

**7.2.25 Customer issues**

For populating customer issues assigned for logged in customer support personnel, Operations application will **invoke GetDataRecord entry point of the DataStore service** to query assigned issues data records from Operations data store for given user ID.

**7.2.26 Knowledge base search**

For issue resolution, customer support personnel will use Operations application UI to **invoke GetDataRecord entry point of the DataStore service** to query resolution data records from Operations data store for specific issue keywords.

**7.2.27 Technical issues**

For populating technical issues forwarded by customer support personnel and assigned to logged in technical support personnel, Operations application will **invoke GetDataRecord entrypoint of the DataStore service** to query assigned issues data records from Operations data store for given user ID.

**7.2.28 Knowledge base update**

For issues resolved, technical support personnel will use Operations application UI to **invoke CreateOrUpdateDataRecord entry point of the DataStore service** to create or update resolution data records in Operations data store for specific issue keywords with associated obtained resolution.

**7.2.29 Basic charts**

On selection of predefined chart dimension, metric and time granularity by business owner, Operations application will **invoke GetChart entry point of the Analytics service** to fetch the chart data to render by running the chart specific query in Analytics data store.

**7.2.30 Analytics charts**

When business owner **configures and save an analytics chart definition** with specific type (comparison, trend, second dimension), dimension, metric and time granularity, Operations application will **invoke ConfigureChart entry point of the Analytics service** to **persist the chart definition** in the Analytics data store and then **invoke GetChart entry point of the Analytics service** **to fetch the chart data** to render by running the chart specific query in Analytics data store.

**7.2.31 Delete chart**

When the business owner **deletes** a **chart**, **DeleteChart entry point of the Analytics service** will be invoked to delete the chart definition from Analytics data store.

**7.2.32 Dashboard for Business Owner**

On login, business owner will be presented with Dashboard. The information on regulatory compliance statuses, business licenses, partner service subscription or contract fees and renewal due dates will be displayed on dashboard which will be fetched using **GetDataRecord entrypoint of the DataStore service** with predefined query attributes.

**7.2.33 Location information**

MappingService will invoke predetermined **Geocoding APIs to get Location lat-long coordinates for given textual location address** and **external Mapping services** (such as Google maps or TomTom maps) **APIs to create the map view with location markers**.

**7.2.34 Route information**

MappingService will invoke predetermined **Routing APIs to get route information for given source and destination locations** and **external Mapping services** (such as Google maps or TomTom maps) **APIs to create the map view with the route information.**

**7.2.35 Cabs in vicinity and map view**

RideManagement service entrypoint GetRideOptions will use driver (cab provider) smartphone GPS coordinates and invoke predetermined **Geospatial Indexing APIs to determine cabs in the vicinity of source location of the ride** and **external Mapping services** (such as Google maps or TomTom maps) **APIs to create the map view with the cabs in vicinity information.**

**7.2.36 Food options**

Food-deliveryProcessing service will **use Food-delivery partner service API to get the food options available at specific food stop if any**.

**7.2.37 Food order**

Food-deliveryProcessing service will **use Food-delivery partner service API to order food at specific food stop**.

**7.2.38 Payment processing**

PaymentProcessing service will **use Payment gateway** APIs (such as PayPal) **to execute the payment transactions**.

**7.2.39 Data record CRUD**

All services will use **DataStoreService entry points CreateOrUpdateDataRecord, GetDataRecord and DeleteDataRecord** for the data record CRUD operations.

**7.2.40 Logging**

All services will use **AuditService WriteLog entry point** for writing log records to Operations data store.

**7.2.41 Phone and e-mail verification**

All applications will use **ValidateOTP and ValidateEmailVerificationLink entry points of ValidationService** for validating the phone and e-mail addresses.

**All-En-Route Deployment Architecture**

**1. Deployment overview**

This **document** depicts **deployment guidelines** for the All-En-Route system. The All-En-Route system will be a **distributed system** with its **user applications communicating over internet with** its **geographically replicated core system**.

The document provides guidelines for deployment with example option of Kubernetes v1.28 cluster deployment in Oracle cloud native environment (OCNE) v1.7. The hardware components table in section 3.7 is also example sizing with example component cardinalities for example option of Kubernetes v1.28 cluster deployment in OCNE v1.7 with generalized working procedures described in section 3.1 and 3.2. The actual cluster and cloud platform for staging and production deployment will be selected after evaluation of available options.

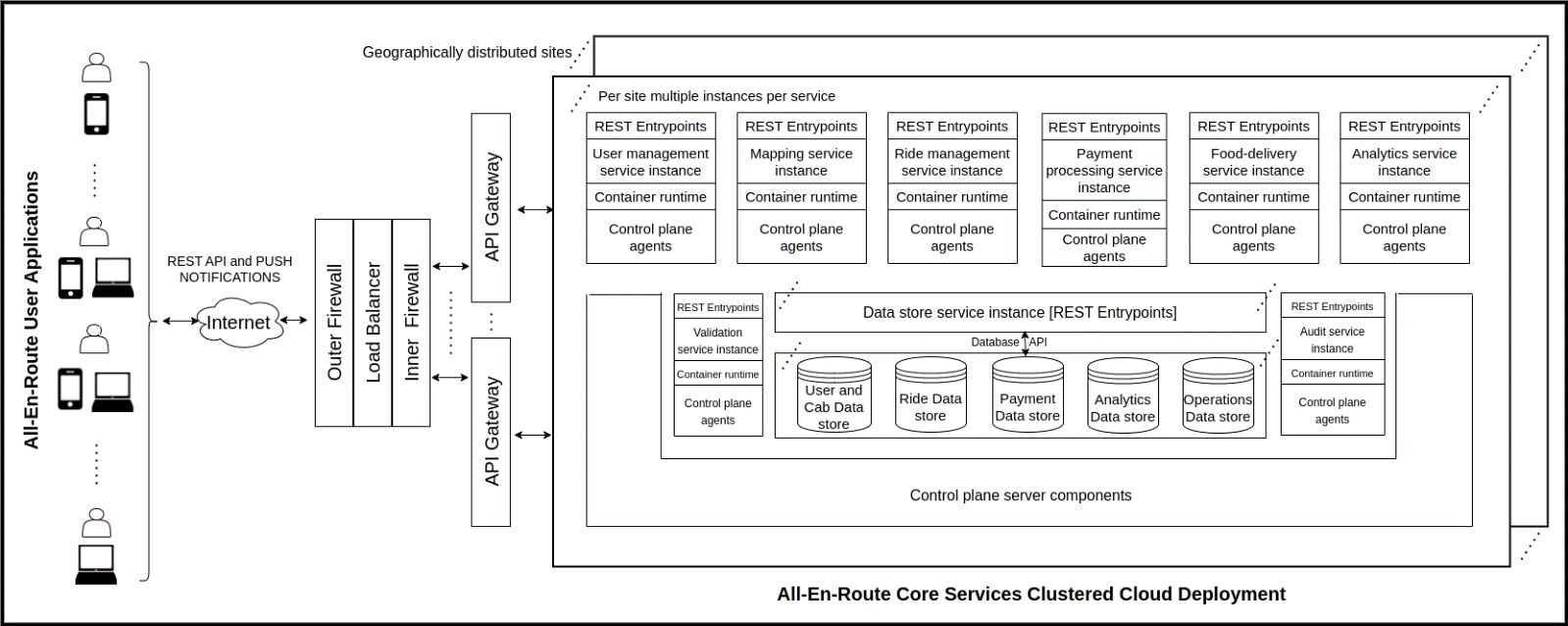
**2. Deployment topology**

The All-En-Route user application requests will be routed through load balancer and API gateway to appropriate All-En-Route core service. The load balancer will reside in dual firewall **DMZ** for optimal security.

The **load balancer** will distribute the traffic among the running instances of **API gateway**. The All-En-Route core service instances will run as containerized native cloud microservice instances to allow resources and cardinality configuration changes based on load once operational. Functionality of maintaining the desired configuration will be provided by the underlying cluster infrastructure.

The core services with respective data stores will run in multiple sites on cloud platform across the country. In each site, configured number of instances of each core service will be hosted as containers in **clustered cloud deployment**. The core services will expose REST API entry points and setup push notifications for client application interactions.

The **data stores** will also be initially sized based on estimated data movement. To support various types of data formats, **NoSQL database instances** will be configured for the data stores.



**Fig 1: All-En-Route high level deployment architecture**

**2.1 System components**

The All-En-Route **user applications** for various roles will be **mobile application** running natively as Android or iOS application on mobile device **and/or** **thin web client application** running on laptop, as described in “Section 2 of Software Architecture” document.

The **core system** will need to be **modular, secured, available anytime anywhere across** the **country** with **low latency, easy to maintain, easy to deploy** and **scalable**. Given these quality attribute requirements, the core system will need to follow service (**microservices**) oriented architecture. **For anticipated scale, operational ease, agility, flexibility** and **optimal resource utilization**, All-En-Route core services **will** need to **run in cluster in cloud environment**.

**2.2 Scale and availability**

Scale wise, **daily millions of active passengers, few hundred thousand of active cab providers,** and **few thousands of active operations staff** will be interacting with the core system. **For low latency,** **core system** will be **geographically replicated** in multiple designated locations. In **each location**, the **core system** will be **deployed in clustered environment in the cloud**. To manage the **scale** requirements, the **maximum resources** and **number of instances per service** will be **configured initially** based **on estimated load** and **later** will be **tuned for actual load**.

The **availability** of the service resources, instances and dependencies configured in the cluster configuration will be **managed and maintained by underlying cluster infrastructure running the microservices**. For instance, if the services are deployed as containers in Kubernetes (k8s) cluster, the k8s control plane will manage and maintain service resources and instances configured for required concurrency and redundancy for high availability. The availability of 99.999% will be targeted using relevant hardware, software, monitoring, redundancy and diagnostic capability.

**3. Deployment components**

|  |  |  |  |
| --- | --- | --- | --- |
|  | **Elements** | | |
| **Components** |  | **Hardware** | **Software** |
| **Compute** | Compute nodes  [Specification: vcpu cores and memory capacity] | Operating system  Cluster software  HTTP/HTTPS server software  NoSQL DBMS software |
| **Storage** | SAN storage  [Specification: IO throughput and storage capacity] | Storage OS  Storage configuration and management software  Backup and restore software  Disaster recovery (replication) software |
| **Network** | Outer and inner firewall  Load balancer  API Gateway  [Specification: Network bandwidth] | Network OS  Firewall software  Load Balancer software  API Gateway software |

**Table 1: Core and platform services (clustered cloud deployment) components per site location**

**3.1 Compute node hardware**

For **production deployment**, per site location, initial compute node hardware sizing for **maximum number of vcpus** will depend on the maximum number of service instances to be run on the node. Initial maximum number of instances for a service will be arrived at using the estimated number of requests to be handled by the service per second at peak load and service throughput in terms of requests per second. Similarly, compute node hardware sizing for **maximum memory capacity** will depend on the estimated peak load data processed and persisted by the service instances running on the node. Per site location, the service instances will also be distributed across the cluster nodes in the cloud to achieve required resilience. The underlying cluster technology will need to provide for this, for instance, k8s inherently supports this.

**Staging deployment** sizing will be scaled down version (if possible and/or required) of production deployment but large enough to cover the concurrency and redundancy and mimic the topological aspects of the production deployment from testing perspective.

For **development**, a single node deployment with one instance of each service will be made available. For **testing**, multinode (three node) deployment with multiple instances of each service will be made available.

**3.2 Compute node software**

Following **software elements** will need to run on each **compute node**:

* **Operating system software**: The compute nodes will run latest default OS distribution supported in preselected cloud platform. The OS installations will need to maintain the OS patch level and security patch levels as and when recommended by the cloud platform.
* **Cluster software**: The cluster software will need to be installed with predecided redundancy for master nodes and rest of the compute nodes will be worker nodes. The worker nodes will host All-En-Route microservices instances allocated by control software running on master nodes. For instance, in k8s cluster deployment, control software will run on predesignated master nodes and container runtime will run on all (master and worker) nodes.
* **HTTP/HTTPS server software**: This will be available in the REST API framework used by the application service.
* **NoSQL DBMS software**: The application data stores will be allocated required storage as configured in the cluster configuration. The NoSQL database to use will be predetermined after evaluating options (such as MongoDB or Cassandra). The DataStoreService will be used to abstract access to the NoSQL data store instances. The cluster infrastructure will provide for DNS based service discovery, IP tables based load balancing (round robin) and traffic distribution to data store replicas. The DataStoreService will interact with data stores using database APIs. The service level access tokens for the data store access will be managed using cluster configurations.

**3.3 Storage hardware**

Multipathed block level **SAN storage** will be used for data stores to provide resiliency and low latency IOs. The storage will be configured with RAID-6 (or at least RAID-5) for better performance and fault tolerance with reasonable resource provisioning.

The **sizing of storage hardware** will be initially based on the estimated data volumes (persisted), backup and replication requirements of the platform and will be tuned to actual volume once operational. The maximum throughput requirements will be determined based on the estimated collective maximum IO per second (IOPS) to be done by all the All-En-Route application services, backup and replication and will be tuned to actual IOPS once operational.

**3.4 Storage software**

Following **software elements** will need to run **on DataStoreService compute nodes**:

* **Storage configuration and management software**: The storage configuration and management software will be needed to configure, allocate and manage the application data stores and provide access control for the same.
* **Backup and restore software**: The data backup and restore software will need to support the Recovery Point Objective (RPO) and Recovery Time Objective (RTO) requirements for All-En-Route system. The periodic backup batch jobs will be configured by the system administrator.
* **Disaster recovery (replication) software**: The storage software will need to optionally support for the replication functionality in case customer plans to configure for disaster recovery. The periodic replication for data stores will be configured by the system administrator.

**3.5 Network hardware**

The core services will be deployed in multiple locations / sites (like capital city of each state). In each site, the internal network or **Virtual Private Cloud (VPC)** will be protected using dual (outer and inner) firewall **Demilitarized Zone (DMZ)**.

* **Outer and inner firewall**: The outer firewall secures ingress from internet to DMZ and inner firewall isolates VPC from DMZ.
* **Load Balancer**: The load balancer will expose the public IP from the DMZ through outer firewall. To avoid being a single point of failure, it will be configured in active standby setup.
* **API Gateway**: The API gateway will act as reverse proxy. For concurrency and to avoid being a single point of failure, it will be deployed with at least two replicas for high availability.

Depending on the initial estimates of the ingress of user application requests and push notifications, **network bandwidth** required will be determined and tuned to actual load once operational.

**3.6 Network software**

Following **software elements** will need to run on network hardware:

* **Network OS**: Each network appliance will run supported default network OS to provide required framework for its designated functionality.
* **Firewall software**: The firewalls enhance security by implementing multiple security checkpoints and controls for incoming and outgoing traffic for VPC.
* **Load Balancer software**: The load balancer will provide traffic distribution with predefined policy such as round robin, weighted round robin or least connections.
* **API Gateway software**: The API gateway will provide API routing and security to VPC against various online threats and attacks, rate limiting, caching, and audit logging functionality.

The networking configuration for VPC will be managed by the cluster infrastructure.

**3.7 Hardware components**

The hardware components table in this section is example sizing with example component cardinalities for example option of Kubernetes v1.28 cluster deployment in OCNE v1.7 with generalized working procedures described in section 3.1 and 3.2.

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **S. No.** | **Component ID** | **Component Name** | **Count** | **Component description** |
| 1 | FW | Firewall | 4 | Outer firewalls (1 active, 1 standby)  Inner firewalls (1 active, 1 standby)  Example: F5 Big-IP iSeries Firewall  Secure VPC from internet |
| 2 | LB | Load balancer | 2 | Load balancer (1 active, 1 standby)  Example: F5 Big-IP iSeries Load balancer  Distribute incoming traffic |
| 3 | AG | API Gateway | 2 | API gateway (2 instances for concurrency)  Example: F5 Big-IP iSeries API Gateway  Route the API calls to appropriate services |
| 4 | CCN-UM  CCN-MS  CCN-RM  CCN-PP  CCN-FD  CCN-AN  CCN-DS  CCN-AS  CCN-VS | Container compute nodes | 30 | Service container compute nodes:  10 service instances for data store service (2 instances per data store)  3 service instances (containers) for each of mapping service (CCN-MS), ride management (CCN-RM), payment processing (CCN-PP), food-delivery (CCN-FD)  2 service instances for each of validation (CCN-VS), analytics (CCN-AN), user management (CCN-UM) and audit (CCN-AS)  Example: Kubernetes worker node in OCNE  Specification:  1 CPU cores (Intel VT-capable CPU), 8GB RAM, 1GB Ethernet NIC  XFS file system (the default file system for Oracle Linux)  15GB hard disk space in the /var directory  XFS mount-point /var/lib/containers  Container compute nodes run application core service instances in the containers |
| 5 | CM | Cluster master nodes | 2 | Example: Kubernetes master node (control plane node) in OCNE  4 CPU cores (Intel VT-capable CPU), 16GB RAM, 1GB Ethernet NIC  XFS file system (the default file system for Oracle Linux)  40GB hard disk space in the /var directory  Cluster master nodes run the cluster control plane components |
| 6 | STOR | Data store storage | 5 | Example: Seagate Exos CORVAULT 4U106 and IBM SAN32C-6 SAN Switch  Storage for data stores |

**Table 2: Hardware components**

**3.8 Application Software Components**

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **S. No.** | **Component ID** | **Component Name** | **Installation H/W Component ID** | **Open source or proprietary?** | **Component description** |
| 1 | UMSC | User management service container | CCN-UM | Proprietary | User management service  User account and profile management |
| 2 | MSC | Mapping service container | CCN-MS | Proprietary | Mapping service  Locations, routes and ETA information |
| 3 | RMSC | Ride management service container | CCN-RM | Proprietary | Ride management service  Ride create, assignment, start, finish, cancel |
| 4 | PPSC | Payment processing service container | CCN-PP | Proprietary | Payment processing service  Payment transaction |
| 5 | FDSC | Food-delivery service  container | CCN-FD | Proprietary | Food-delivery service  Food stops and option listing and order placement |
| 6 | ANSC | Analytics service container | CCN-AN | Proprietary | Analytics service  Basic and analytics charts configuration and management |
| 7 | DSSC | Data store service container | CCN-DS | Proprietary | Data store service  Data store, retrieve, update and delete |
| 8 | ASC | Audit service container | CCN-AS | Proprietary | Audit service  System activities logging |
| 9 | VSC | Validation service container | CCN-VS | Proprietary | Validation service  Phone, email validations |

**Table 3: Application Software Components (for each deployment unit)**

**3.9 System Software Components**

The system software table in this section is listing of software for example option of Kubernetes v1.28 cluster deployment in OCNE v1.7 with generalized working procedures described in section 3.1 and 3.2.

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **S. No.** | **Component ID** | **Component Name** | **Installation H/W Component ID** | **Open source or proprietary?** | **Component description** |
| 1 | CSS | Container system software | CCN-UM  CCN-MS  CCN-RM  CCN-PP  CCN-FD  CCN-AN  CCN-DS  CCN-AS  CCN-VS | Open source | HTTP/HTTPS server software (for REST API entry points)  K8S Container runtime  K8S Cluster control plane agents  Oracle Linux v9  Provide the container runtime and control plane agent side functionality |
| 2 | FWS | Firewall software | FW | Proprietary | F5 BIG-IP software stack on F5 TMOS operating system  Secure VPC from internet |
| 3 | LBS | Load balancer software | LB | Proprietary | F5 BIG-IP software stack on F5 TMOS operating system  Distribute incoming traffic |
| 4 | AGS | API gateway software | AG | Proprietary | F5 BIG-IP software stack on F5 TMOS operating system  Route the calls to appropriate core service |
| 5 | CMS | Cluster master node software | CM | Open source | Kubernetes v1.28  Cluster infrastructure |
| 6 | STORSW | Data store storage | STOR | Proprietary | Example: Seagate Exos CORVAULT 4U106 drivers and management software and IBM SAN32C-6 SAN Switch drivers and management software  SAN configuration and data store access |

**Table 4: System (Infrastructure) Software Components**

**4. Deployment process**

The All-En-Route software **installable, upgrade and patches** (fixes) will be made available in the release software images repository for the system administrator to download. The software images will be downloaded and installed as per the All-En-Route **installation guide** following the documented **best practices**.

The installations, upgrades and patches will be **first applied in staging environment**. After testing it in staging for stipulated time, it will be **applied to production environment**.

The **best practices** for **setting up cluster configuration for required concurrency and redundancy** will be followed. The services will be upgraded in rolling fashion, by upgrading one instance at a time.

**5. Security considerations**

The user **RBAC** will be setup for predefined user accounts, users created by online registration or when system administrator configures system users. The data **encryption** will also be configured for user and cab data store and payment data store for securing data at rest. All the REST APIs will use https protocol for secured interaction and transactions. The security in the VPC will use **security tokens** for authentication and authorization for service entry point invocations and service-to-data-store communication.

All the users’ personally identifiable information (**PII**) and **financial transactions** will be **encrypted** and **passwords** will be **one-way-hashed** in user application itself before being transferred on-the-wire and persisted at-rest in data stores in VPC for **security** and **privacy** reasons.

**6. Monitoring and logging**

The system administrator will setup cloud native or opensource (such as Prometheus) **monitoring tools for monitoring system health, performance and capacity**. All the components will **log the activities using common audit service** to facilitate system auditing and **diagnostics** when needed.

**7. Disaster recovery and backup**

The **data stores** will be **backed up periodically** and **replicated** to designated centralized locations **for disaster recovery** reasons. **Analytics functionality** will **operate** in central locations **on full data** for the **analytics across all sites**. The **knowledge base data** will also be **replicated** and merged across all sites for optimal issue resolutions.

The **periodic backup job**s and configurations for regular data backups will be setup by system administrator. The **restore procedures** and **disaster recovery strategies** and procedures for recovering from system failures or disasters will also be setup and configured as automated jobs to be run on demand or triggered on an event basis by system administrator.

**References**

**[1]** [The Zachman Framework](https://zachman-feac.com/the-framework-for-enterprise-architecture-background-description-and-utility)

**[2]** [India Radio/Online Taxi Services Market Competition Forecast and Opportunities, 2029](https://www.researchandmarkets.com/reports/4623150/india-radioonline-taxi-services-market)

**[3]** [Software Systems Architecture: Working With Stakeholders Using Viewpoints and Perspectives](https://www.viewpoints-and-perspectives.info/)

**[4]** [The Complete List of Indian Cab Ecosystem, features & technologies](https://medium.com/@siddharth.parakh/the-complete-list-of-indian-cab-ecosystem-features-technologies-672d7760a569)

**[5]** [Fundamentals of Software Architecture: An Engineering Approach](https://www.oreilly.com/library/view/fundamentals-of-software/9781663728357/)

**[6]** [Business Analytics (BA): Everything You Need to Know](https://www.asmibmr.edu.in/blog/business-analytics-ba-everything-you-need-to-know/#5_Google_Analytics)

**[7]** [Business Intelligence Dashboard: All You Need to Know](https://www.knowledgehut.com/blog/business-intelligence-and-visualization/business-intelligence-dashboard)

**[8]** [What’s A Sysadmin? The System Administrator Role Explained](https://www.splunk.com/en_us/blog/learn/system-administrator-sysadmin-role.html)

**[9]** [Technical support](https://en.wikipedia.org/wiki/Technical_support)

**[10]** [Service-Oriented Architecture: Concepts, Technology, and Design By Thomas Erl](https://www.thomaserl.com/book/service-oriented-architecture-concepts-technology-and-design/overview/index.html)

**[11]** [Software Architecture Patterns by Mark Richards](https://www.oreilly.com/library/view/software-architecture-patterns/9781098134280/)

**[12]** [The Complete List of Indian Cab Ecosystem, features & technologies](https://medium.com/@siddharth.parakh/the-complete-list-of-indian-cab-ecosystem-features-technologies-672d7760a569)

**[13]** [Business Intelligence Dashboard: All You Need to Know](https://www.knowledgehut.com/blog/business-intelligence-and-visualization/business-intelligence-dashboard)

**[14]** [Long Polling: A easily implemented concept](https://pragyasapkota.medium.com/long-polling-a-easily-implemented-concept-16efee1ff587)

**[15]** [Geocoding API overview](https://developers.google.com/maps/documentation/geocoding/overview)

**[16]** [Routes API](https://developers.google.com/maps/documentation/routes)

**[17]** [Leveraging your GPS data using Geospatial analytics](https://medium.com/@noahdata7/leveraging-your-gps-data-using-geospatial-analytics-d1865cf6bb7)

**[18]** [Software Systems Architecture](https://www.viewpoints-and-perspectives.info/)

**[19]** [Microservices Architecture](http://medium.com/design-microservices-architecture-with-patterns/microservices-architecture-2bec9da7d42a)

**[20]** [Deployment Patterns in Microservices Architecture](https://www.developer.com/design/deployment-patterns-microservices/)

**[21]** [Oracle Cloud Native Environment](https://docs.oracle.com/en/operating-systems/olcne/1.7/start/#Oracle-Cloud-Native-Environment)

**[22]** [BIG-IP System](https://www.f5.com/pdf/data-sheet/big-ip-platforms-datasheet.pdf)

**[23]** [Exos E 4U106](https://www.seagate.com/in/en/products/storage/data-storage-systems/jbod/exos-e-4u106/)

**[24]** [IBM Storage Networking SAN32C-6 Fabric Switch](https://www.ibm.com/products/san32c-6)

**[25]** [Microservices architecture on Azure Kubernetes Service](https://learn.microsoft.com/en-us/azure/architecture/reference-architectures/containers/aks-microservices/aks-microservices)

**[26]** [Understanding RPO and RTO](https://www.druva.com/blog/understanding-rpo-and-rto)

**[27]** [Data replication](https://www.techtarget.com/searchdisasterrecovery/definition/data-replication)